Visual Programming Language for the Tacit Subset of J Programming Language

Nouman Tariq

Agenda

- Introduction
- Problem Statement
- Related Works
- J Programming Language
- Blockly
- Visual J
- Evaluation
- Conclusion

Introduction

- Use of graphical notation for the purpose of programming
- Diagrams
  - *"A figure drawn in such a manner that the geometrical relations between the parts of the figure illustrate relations between other objects"*
  - *Data Flow Diagram*
  - *Flowcharts*

Problem Statement

- Textual languages provide richness of expression and flexibility
- Visual languages suffer from a number of barriers
  - Globally understandable icons
  - Icons to express abstract notions
  - Abstraction
- Is it possible to create a visual programming language that overcomes these challenges?

Motivation

- Reduced learning curve
- Better visualization
- Enabling parallel processing
- Freedom of expression
- Suitability of functional programming

Aims and Objectives

- Easy to learn
- Less focus on semantics
- Fully functional general purpose programming environment
- Parallelization

Early Works

- Xerox created first GUI
- Sutherland (1966)
- Smalltalk (Goldberg, 1983)
- PICT (Glinert, 1985)
- Blox (Glinert, 1986)
- Limited by the hardware and software limitations of the time

HI-VISUAL

- An Iconic programming language
- Icons represented objects in the system
  - Combined data and functions
  - Internal View: Concept, Substance, Messages (Acceptable and Transmittable)
  - External View: The iconic representation
- Active icons vs. Passive icons

HI-VISUAL Contd.

- Generic framework
- Tightly coupled with the specific environment being modelled
- Difficult to represent abstract objects

Visual Logic Programming (VLP)

- Developed in 1996
- Two guiding design principles
  - Base the icons on experimental studies
  - Combining text and icons for greater flexibility
- Difficulty in expressing first order predicate logic graphically
- Good handling of input parameters

VLCC

- Visual Language Compiler-Compiler (VLCC)
- Allows language designers to design a visual language
- Generates the IDE and visual editor for the designed language
- Provides no guidelines for design

VisaVis

- A functional visual programming language
- Uses FFP (Formal system for Functional Programming) as the evaluation engine
- Uses meta icons for every order function augmented with colours and shadows
- Difficult to use for novice programmer
- Use of FFP limits general purpose programming

Blockly

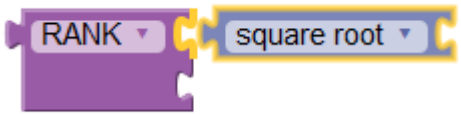
- Developed by Google
- Open source
- A generic visual language
- Uses jigsaw puzzle for the visual metaphor
- Web based

Language Philosophy

- Meant for novice users
- A learning tool
- One based indexing
- Variable names
- Abstraction using high level blocks
- Programming language independence

Blocks

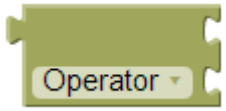
- The basic building block for Blockly
- Protrusions = Output
- Sockets = Inputs

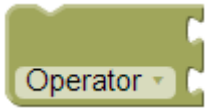


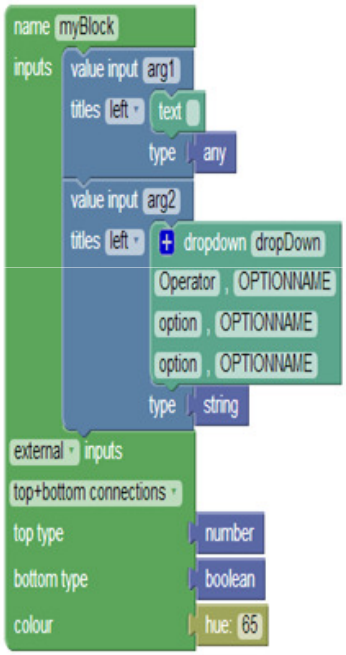
Defining Blocks

```
Blockly.Language.myblock = {
  helpUrl: 'http://www.example.com/',
  init: function() {
    this.setColour(65);
    this.appendValueInput("arg1")
        .setCheck("null")
        .appendTitle("");
    this.appendValueInput("arg2")
        .setCheck("String")
        .appendTitle(new Blockly.FieldDropdown([["Operator", "OPTIONNAME"], ["option", "OPTIONNAME"], ["option", "OPTIONNAME"]]), "dropDown");
    this.setPreviousStatement(true, "Number");
    this.setNextStatement(true, "Boolean");
    this.setTooltip('');
  }
};
```

Defining Blocks (Contd.)







J Programming Language

- Array manipulation notation called Iverson Notation
- Predecessor: APL
- General purpose language
- Array based language
- Functional Programming

Array Based Programming

- Every thing is an array
- Scalars as a special case for arrays
- Eliminates need for looping constructs

Structure of J

- Based on English grammar
- Nouns
- Verbs
  - Monads
  - Dyads
  - More than two parameters
- Adverbs
- Conjunctions

Order of Evaluation

- Two primary rules
  - All verbs have equal precedence
  - Order of evaluation is from right to left
- Fragments
- Differentiating between monads and dyads
- x * y + z
- x + _y
- x var 5

Tacit Programming

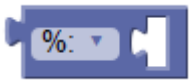
- The most compact form of J code
- Allows combining J constructs with out specifying information about arguments
  - Avg =: +/ % #
  - Avg 1 2 3 4 5

Visual J

- Customize Blockly to create a visual program that can translate to J code
- Provide graphical notations for J constructs
  - Monads
  - Dyads
  - Adverbs
  - Conjunctions
  - Noun

Block for Monad

- Requirements
  - Needs one argument
  - Needs a way to specify monad
  - Needs to specify output
- Solution



Block for Dyad

- Requirement
  - Two input parameters
  - Selection of verb
  - Output
- Solution

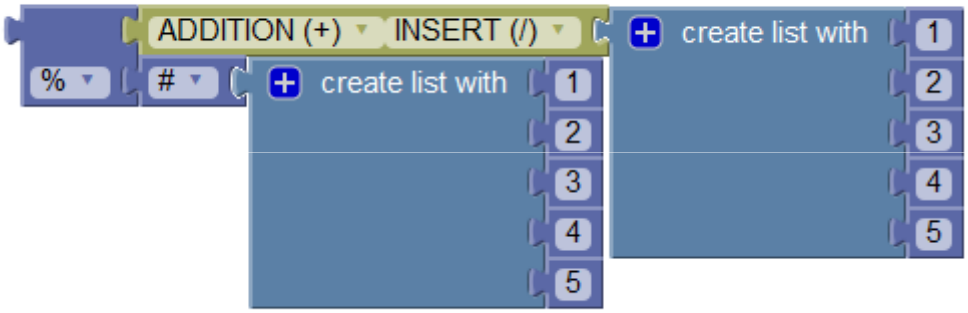
Block for Adverb

- Requirements
  - Requires one verb
  - Allow the user to pass arguments to the verb
  - Specify output
- Solution

Block for Conjunction

- Requirements
  - Requires two verbs
  - Provide inputs for those verbs
  - Provide output
- Solution

Visual J



+/ 1 2 3 4 5 % # 1 2 3 4 5

Evaluation

- Possible criterions
  - Character count
  - Lines of Code
  - Conditional Statements – McCabe
  - Operators and Operands – Halstead
- Halstead metric for Visual J
  - # of operands + # of operators

Results

- Textual J and Visual J provide the same complexity!
- Results are as expected

Threats to Validity

- Scalability
- Time to program
- Burden of syntax

Conclusion

- Visual J does not add additional complexity to code
- Useable by novice programmers
- Useable by expert programmer
- Does not restrict the programmer's freedom of expression

Future Work

- Automatic parallelization
- Implement trains
- Code reusability
- Code execution

Thank you

Nouman Tariq
NUI, Maynooth
numantariq@gmail.com